FOTREZ

Test Results

NETWORK MONTERING –

sonic_avik@fedora:~/NetworkMonitorApp/src/dist ~/NetworkMonitorApp/src/dist

sonic_avik@fedora:~$ python /home/sonic_avik/NetworkMonitorApp/src/dist/NetworkMonitor

File "/home/sonic_avik/NetworkMonitorApp/src/dist/NetworkMonitor", line 1 ELF

SyntaxError: source code cannot contain null bytes

sonic_avik@fedora:~$ cd /home/sonic_avik/NetworkMonitorApp/src/dist

sonic_avik@fedora:~/NetworkMonitorApp/src/dist$ python NetworkMonitor.exe |||X

python: can't open file '/home/sonic_avik/NetworkMonitorApp/src/dist/NetworkMonitor.exe': [Er rno 2] No such file or directory

sonic_avik@fedora:~/NetworkMonitorApp/src/dist$ '/home/sonic_avik/NetworkMonitorApp/src/dist/

NetworkMonitor'

Monitoring started.

Exception in thread Thread-2 (start_sniffing):

Monitoring started.

Traceback (most recent call last):

File "threading.py", line 1041, in _bootstrap_inner

File "threading.py", line 992, in run

File "src/network_monitor_gui.py", line 125, in start_sniffing

File "scapy/sendrecv.py", line 1424, in sniff

File "scapy/sendrecv.py", line 1273, in _run

File "scapy/arch/linux/__init__.py", line 218, in __init__

File "socket.py", line 233, in __init__

PermissionError: [Errno 1] Operation not permitted

Monitoring stopped.

Monitoring stopped.

Monitoring started.

Monitoring started.

Exception in thread Thread-5 (start_sniffing):

Traceback (most recent call last):

File "threading.py", line 1041, in _bootstrap_inner

File "threading.py", line 992, in run

File "src/network_monitor_gui.py", line 125, in start sniffing

File "scapy/sendrecv.py", line 1424, in sniff

File "scapy/sendrecv.py", line 1273, in _run

File "scapy/arch/linux/__init__.py", line 218, in __init__

File "socket.py", line 233, in __init__

PermissionError: [Errno 1] Operation not permitted

Monitoring stopped.

Monitoring stopped.

sonic_avik@fedora:~/NetworkMonitorApp/src/dist$

sonic_avik@fedora:~/NetworkMonitorApp/src/dist$

BOOTSTRAP DATA SNIFFING –

sonic_avik@fedora:~/NetworkMonitorApp/src/dist — /home/sonic_avik/Network...

~/NetworkMonitorApp/src/dist

sonic_avik@fedora:~$ python /home/sonic_avik/NetworkMonitorApp/src/dist/NetworkMonitor

File "/home/sonic_avik/NetworkMonitorApp/src/dist/NetworkMonitor", line 1 ELF

SyntaxError: source code cannot contain null bytes

sonic_avik@fedora:~$ cd /home/sonic_avik/NetworkMonitorApp/src/dist

sonic_avik@fedora:~/NetworkMonitorApp/src/dist$ python NetworkMonitor.exe

python: can't open file '/home/sonic_avik/NetworkMonitorApp/src/dist/NetworkMonitor.exe': [Er rno 2] No such file or directory

sonic_avik@fedora:~/NetworkMonitorApp/src/dist$ '/home/sonic_avik/NetworkMonitorApp/src/dist/

NetworkMonitor'

Monitoring started.

Exception in thread Thread-2 (start_sniffing):

Monitoring started.

Traceback (most recent call last):

File "threading.py", line 1041, in _bootstrap_inner

FOTREZ

File "threading.py", line 992, in run

File "src/network_monitor_gui.py", line 125, in start_sniffing

File "scapy/sendrecv.py", line 1424, in sniff

File "scapy/sendrecv.py", line 1273, in _run

File "scapy/arch/linux/__init__.py", line 218, in __init__

File "socket.py", line 233, in __init__

PermissionError: [Errno 1] Operation not permitted

FOTREZ

HARDCOPY OF MAIN GUI CODE

```
import os
import tkinter as tk
import threading
import time
import scapy.all as scapy
from collections import deque, Counter
import smtplib
from email.mime.text import MIMEText
import logging
import numpy as np
from flask import Flask, jsonify
# Ensure the logs directory exists
log_dir = os.path.join(os.getcwd(), "logs")
os.makedirs(log_dir, exist_ok=True)
# Set up logging
log_file = os.path.join(log_dir, "network_monitor.log")
logging.basicConfig(filename=log_file, level=logging.INFO)
# Configuration (Email)
SENDER_EMAIL = "soniccolab764@gmail.com"
SENDER_PASSWORD = "@VIKmurmu1234"  # Use App Passwords for security
RECEIVER_EMAIL = "sonicdump764@gmail.com"
# Configuration (Network Monitoring)
INTERFACE = "eno1"  # Replace with your network interface
PACKET_WINDOW_SIZE = 100
ALERT_THRESHOLD_PACKETS = 1000
ALERT_THRESHOLD_UNIQUE_IPS = 50
ALERT_THRESHOLD_PROTOCOL_PERCENTAGE = 80
# Data structures for packet analysis
```

FOTREZ

```
packet_counts = deque(maxlen=PACKET_WINDOW_SIZE)
protocol_counts = Counter()
packet_sizes = np.array([])
source_ips = set()
# Global control variables
bot_running = False
sniff_thread = None
analysis_thread = None
def packet_callback(packet):
    """Processes captured packets and updates counters."""
    global packet_sizes
    if packet.haslayer(scapy.IP):
        packet_counts.append(packet)
        packet_sizes = np.append(packet_sizes, len(packet))
        source_ips.add(packet[scapy.IP].src)
        if packet.haslayer(scapy.TCP):
            protocol_counts["TCP"] += 1
        elif packet.haslayer(scapy.UDP):
            protocol_counts["UDP"] += 1
        elif packet.haslayer(scapy.ICMP):
            protocol_counts["ICMP"] += 1

def send_email_alert(subject, message):
    """Sends email alerts."""
    msg = MIMEText(message)
    msg["Subject"] = subject
    msg["From"] = SENDER_EMAIL
    msg["To"] = RECEIVER_EMAIL
    try:
        with smtplib.SMTP_SSL("smtp.gmail.com", 465) as server:
            server.login(SENDER_EMAIL, SENDER_PASSWORD)
```

FOTREZ

```
            server.sendmail(SENDER_EMAIL, RECEIVER_EMAIL, msg.as_string())
        logging.info(f"Alert sent: {subject} - {message}")
    except Exception as e:
        logging.error(f"Failed to send email: {e}")
def analyze_traffic():
    """Analyzes network traffic and detects anomalies."""
    global bot_running, packet_sizes
    while bot_running:
        time.sleep(1)  # Analyze every second
        packets_per_second = len(packet_counts)
        unique_ips_count = len(source_ips)
        # Protocol Analysis
        total_packets = sum(protocol_counts.values())
        dominant_protocol = None
        dominant_percentage = 0
        if total_packets > 0:
            for protocol, count in protocol_counts.items():
                percentage = (count / total_packets) * 100
                print(f"{protocol}: {percentage:.2f}%")
                if percentage > dominant_percentage:
                    dominant_percentage = percentage
                    dominant_protocol = protocol
        # Packet Size Analysis
        if packet_sizes.size > 0:  # Ensure packet_sizes is not empty
            avg_size = np.mean(packet_sizes)
            print(f"Average packet size: {avg_size:.2f} bytes")
        # Alert Conditions
        if packets_per_second > ALERT_THRESHOLD_PACKETS or unique_ips_count >
ALERT_THRESHOLD_UNIQUE_IPS or (
            dominant_percentage > ALERT_THRESHOLD_PROTOCOL_PERCENTAGE and dominant_protocol is not
None
        ):
```

FOTREZ

```
            send_email_alert(
                "DDoS Attack Alert",
                f"A potential DDoS attack detected!\n"
                f"Packets per second: {packets_per_second}\n"
                f"Unique IPs: {unique_ips_count}\n"
                f"Dominant protocol: {dominant_protocol if dominant_protocol else 'N/A'}"
            )
        # Clear old data for the next window
        protocol_counts.clear()
        packet_sizes = np.array([])  # Reset packet_sizes correctly
        source_ips.clear()
def start_sniffing():
    """Starts network packet sniffing."""
    global bot_running
    bot_running = True
    scapy.sniff(iface=INTERFACE, prn=packet_callback, store=0)
def start_bot():
    """Starts the monitoring bot."""
    global bot_running, sniff_thread, analysis_thread
    bot_running = True
    sniff_thread = threading.Thread(target=start_sniffing)
    analysis_thread = threading.Thread(target=analyze_traffic)
    sniff_thread.start()
    analysis_thread.start()
    print("Monitoring started.")
def stop_bot():
    """Stops the monitoring bot."""
    global bot_running
    bot_running = False
    if sniff_thread and analysis_thread:
        sniff_thread.join()
```

```
        analysis_thread.join()
    print("Monitoring stopped.")
# Tkinter GUI Application
class App:
    def __init__(self, root):
        self.root = root
        self.root.title("Network Monitoring")
        self.root.geometry("400x300")  # Window size
  # Process state tracking
        self.process_running = False
    # Create a frame for layout
        self.frame = tk.Frame(root)
        self.frame.pack(expand=True, fill=tk.BOTH)
 # Status label
        self.status_label = tk.Label(self.frame, text="Status: Idle", font=("Arial", 14))
        self.status_label.pack(pady=20)
 # Start Button
        self.start_button = tk.Button(self.frame, text="Start", command=self.start_process, width=15, height=2)
        self.start_button.pack(pady=10)
  # Stop Button
        self.stop_button = tk.Button(self.frame, text="Stop", command=self.stop_process, width=15, height=2)
        self.stop_button.pack(pady=10)
   # Terminate Button
      self.terminate_button = tk.Button(self.frame, text="Terminate", command=self.terminate_process,
width=15, height=2)
        self.terminate_button.pack(pady=10)
 def start_process(self):
        """Starts the network monitoring process."""
        if not self.process_running:
            self.process_running = True
            self.monitoring_thread = threading.Thread(target=start_bot)
```

```
        self.monitoring_thread.start()
        self.update_status("Status: Monitoring")
        print("Monitoring started.")

    def stop_process(self):
        """Stops the network monitoring process gracefully."""
        if self.process_running:
            stop_bot()
            self.process_running = False
            self.update_status("Status: Stopped")
            print("Monitoring stopped.")
    def terminate_process(self):
        """Terminates the monitoring process forcefully."""
        if self.process_running:
            self.process_running = False
            global bot_running
            bot_running = False
            self.update_status("Status: Terminated")
            print("Monitoring forcibly terminated.")
    def update_status(self, message):
        """Updates the status label in the GUI."""
        self.status_label.config(text=message)
# Run the GUI Application
if __name__ == "__main__":
    root = tk.Tk()
    app = App(root)
    root.mainloop()
```